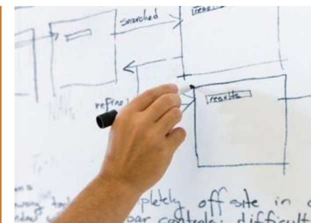





MOCA

24 de febrero de 2013



Agenda

- Introducción
- Historia
- Componentes
- Elementos de la arquitectura MOCA
- MOCA Registry
  - Windows
  - Non-Windows
- Descripción del Ambiente
- Elementos de un Comando
- Creación de Comandos
  - Admón. Comandos de Servidor
  - Creación Manual de Archivo de Comando
  - Sintaxis Local
  - Reemplazo de Variables
  - Operadores
- Comandos Útiles

RedPrairie®



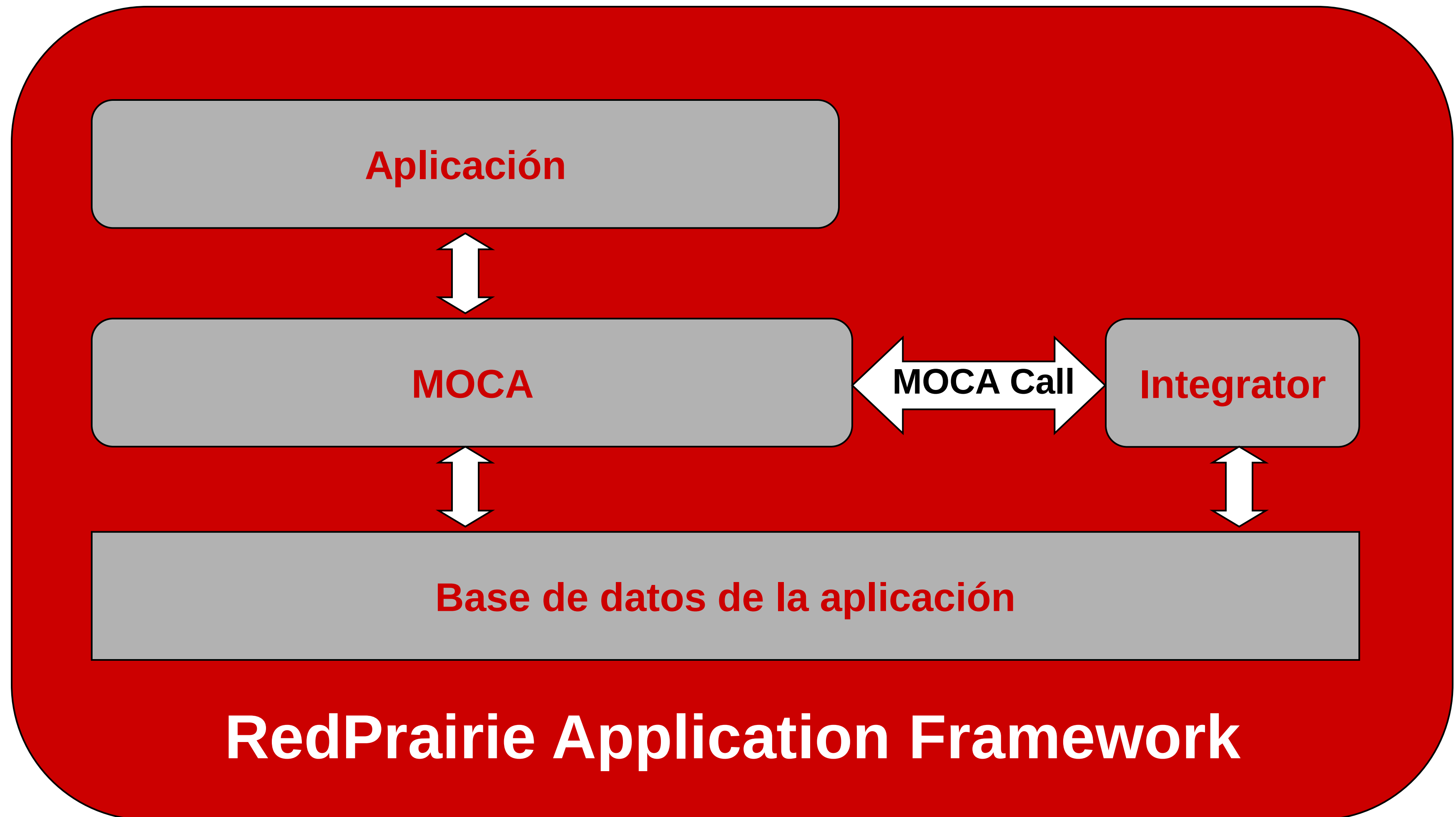
MOCA: Introducción

¿Qué es MOCA?

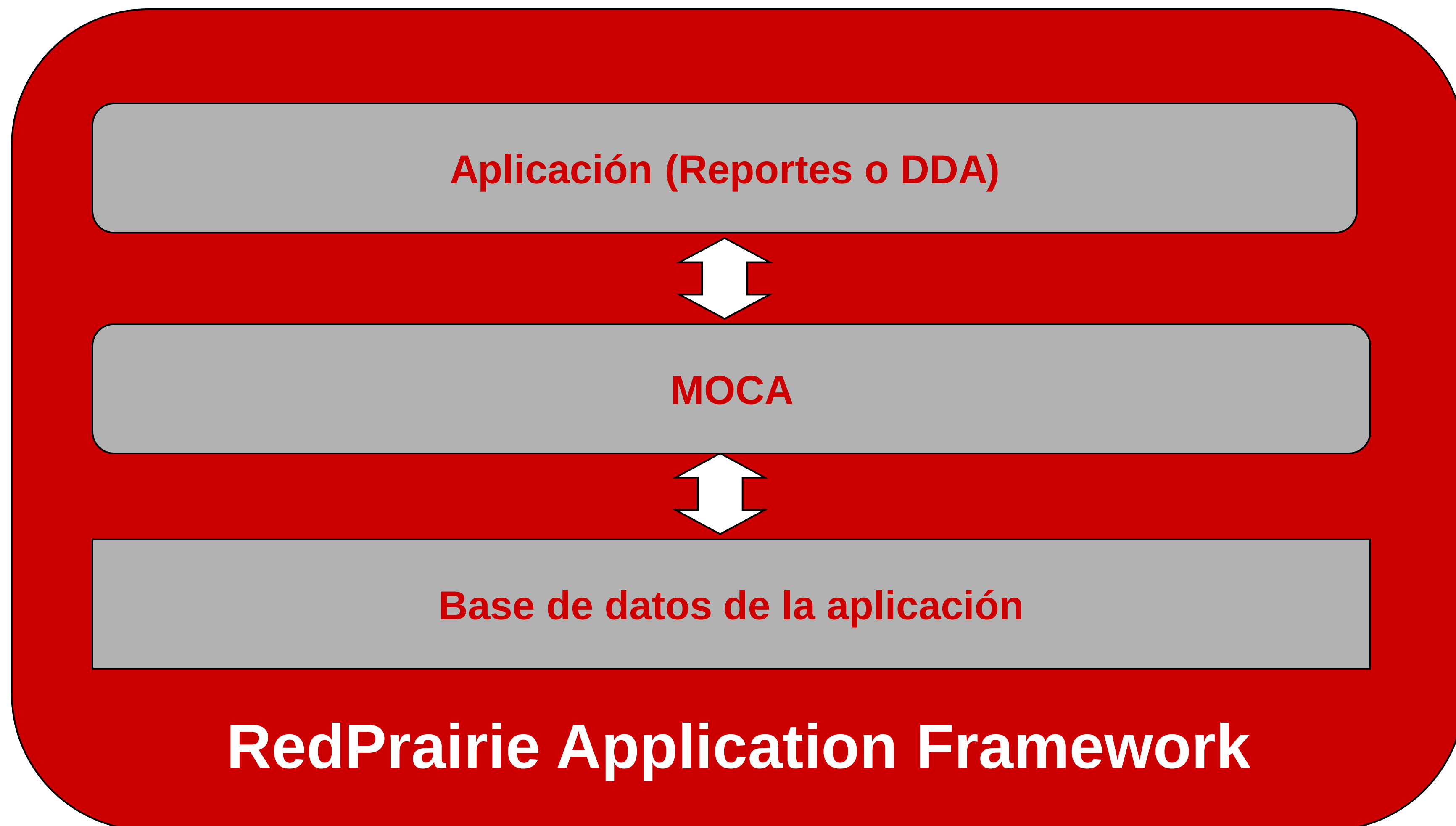
- MOCA es un entorno de RedPrairie diseñado para crear aplicaciones basadas en componentes.
  - La misión de MOCA es ofrecer a los desarrolladores una infraestructura técnica que permita y promueva la interoperabilidad de componentes de una manera simple, efectiva y libre de implementaciones.
  - MOCA ofrece un método para definición, invocación y licenciamiento de componentes así como librerías de soporte para ayudar en la creación eficiente de aplicaciones.

RedPrairie

Application Framework



Application Framework



¿Por qué MOCA?

- Clientes
  - > Los clientes son aplicaciones cuya principal responsabilidad es la salida de datos, interactuar con el usuario e invocar componentes en un servidor.
  - > Toda la funcionalidad es delegada a los componentes que existen en el servidor
  - > Esto permite que las aplicaciones MOCA sean desarrolladas mucho más rápidamente que una aplicación cliente que contiene lógica y funcionalidad adicionales.

RedPrairie

¿Por qué MOCA?

- Facil de Personalizar
  - > MOCA ofrece una arquitectura para personalizar fácilmente una aplicación sin tener que reescribir numerosas líneas de código.
  - > Esto hace mucho más fácil añadir o cambiar la funcionalidad en un producto estándar.

RedPrairie

¿Por qué MOCA?

- Actualizaciones de productos estándar
  - > La arquitectura de MOCA hace más fácil la aplicación de parches y/o actualizaciones al producto estándar ya que todas las personalizaciones residen en diferentes componentes que están fuera del alcance del producto estándar.

RedPrairie



MOCA: Historia

Historia

- Moca fue originalmente derivado de la librería del servidor de DLx® Warehouse/D.
  - La primera liberación fue en 1994.
- La librería del servidor DLx® Warehouse/D soportaba gran parte de la misma funcionalidad que Moca soporta actualmente.

RedPrairie

Historia

- Los componentes pueden ser desarrollados usando sintaxis local y componentes C.
- Los componentes pueden tener argumentos y pueden tener triggers que sean otros comando o sintaxis SQL.



MOCA: Componentes

Componentes

- Un componente es una implementación de funcionalidad capaz de ejecutarse individualmente y que tiene una interfaz conocida.
  - Los usuarios de un componente no necesitan saber detalles de la implementación que se maneja “detrás de cámaras”.
  - Solo necesitan saber que entradas (argumentos) y salidas (datos publicados) son necesarias y que funcionalidad ofrece el componente.

RedPrairie

Componentes

- Combinando un componente, que provee una funcionalidad específica con otro u otros componentes que proveen funcionalidades específicas, se pueden realizar tareas más complejas fácilmente

Niveles de componentes

- Todos los comandos son almacenados en varios niveles de componentes
  - > Cada nivel tiene un numero especifico de secuencia.
  - > El servidor de procesos busca el comando requerido en orden descendente basándose en el numero de secuencia.
- Esto es lo que permite una fácil implementación de nuevos comandos para Redprairie y sus clientes.

RedPrairie

Niveles de Secuencia en los Componentes

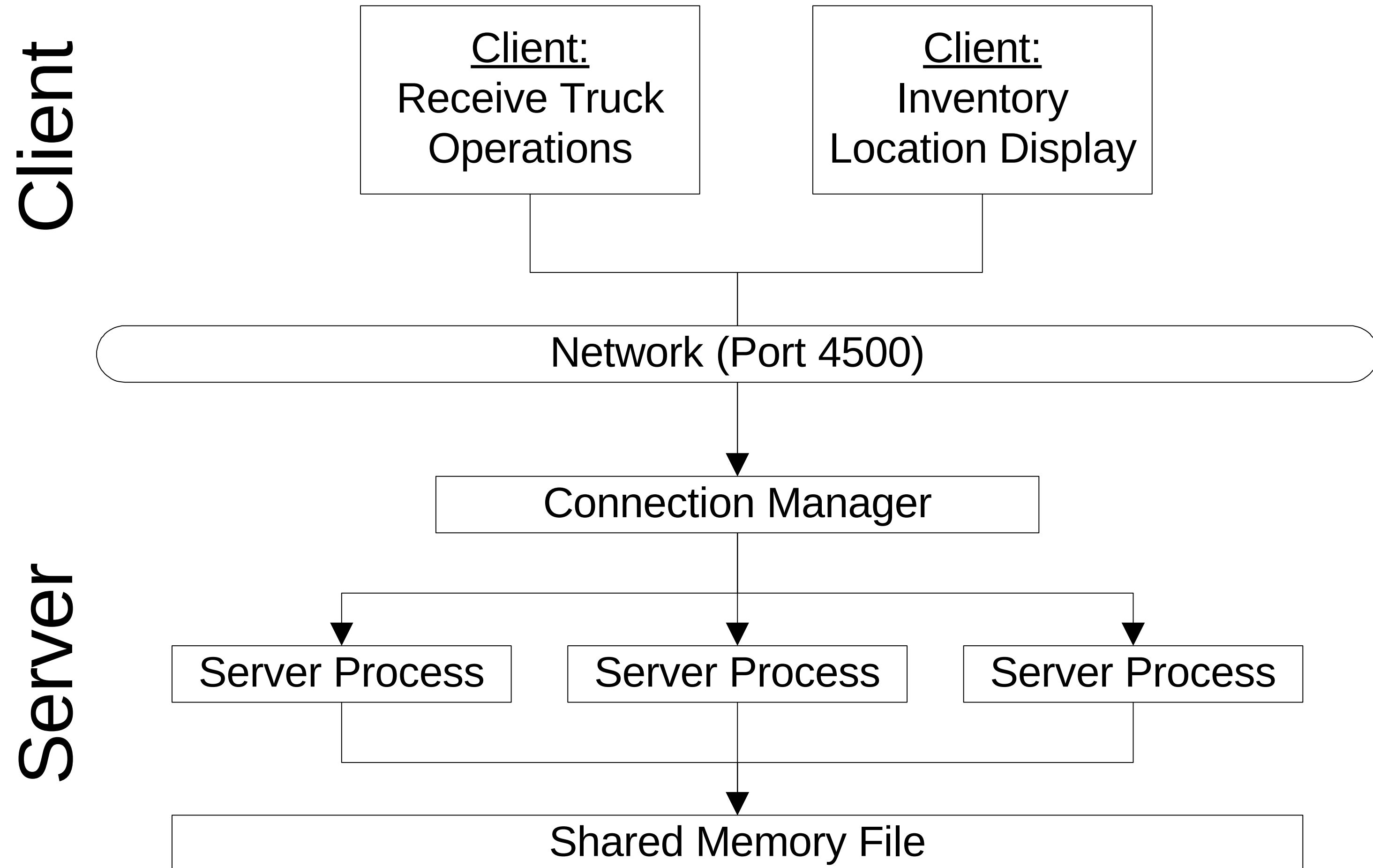
Level	Sequence
Usr (User Customizations)	9000
Var (RedPrairie Customizations)	8000
Specific Product (i.e. Warehouse Management)	300 – 3500
SAL (Shared Application Library)	200 – 299
MCS (McHugh Component System)	100 – 149
MOCA (McHugh Open Component Architecture)	0 – 99

RedPrairie



MOCA: Elementos de la arquitectura MOCA

Arquitectura



Arquitectura – Connection Manager

- El Connection Manager es el módulo principal responsable de administrar las solicitudes de conexión entrantes en un puerto específico.
  - Las solicitudes de conexión pueden ser originadas desde el cliente Windows, la Terminal de RF, MSQL, Integrator, o cualquier otro sistema MOCA habilitado.
- Del lado del servidor, estos clientes tienen la misma estructura y ejecutan los mismos tipos de comandos exactamente de la misma manera.

RedPrairie

Arquitectura –Connection Manager

- El connection Manager es el punto inicial para toda actividad MOCA
  - > Todos los clientes se conectan por el connection manager.
  - > Cuando un cliente hace una solicitud, el connection Manager detecta la actividad en el puerto y pasa el requerimiento a un MOCA Server Process.

Arquitectura – MOCA Server Process

- El MOCA Server Process es el módulo de trabajo actual.
- Mientras el connection manager controla principalmente el número de procesos del servidor y el tráfico relacionado con esos procesos, el MOCA Server Process realiza todo el procesamiento solicitado por el cliente

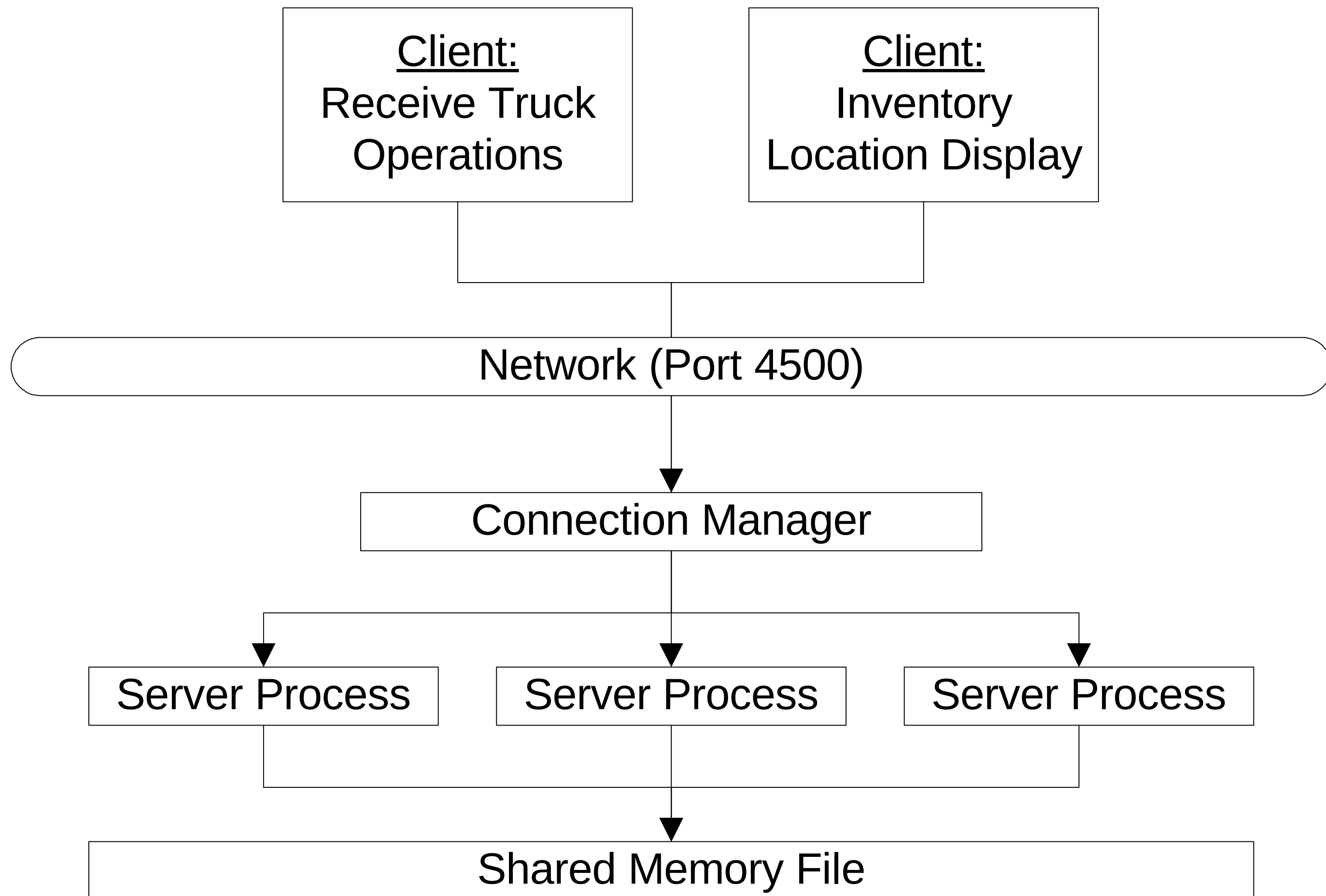
RedPrairie

Arquitectura – Archivo de Memoria

- El archivo de memoria contiene la configuración actual de los comandos y triggers del sistema.
  - Los cambios en la configuración no tienen efecto hasta que el archivo de memoria es recargado.

RedPrairie

Arquitectura



Analogía

- Repartidores – Moca Server Process
- Operador – Connection Manager
- Orden – Client Request
  - > Donde cualquier Repartidor puede procesar cualquier Orden
  - > Toda la comunicación es atreves del Operador
  - > El Operador asigna un Repartidor disponible
    - Si ninguno esta disponible, crea un nuevo Repartidor
    - Si esta ocioso, elimina los Repartidores innecesarios
  - > El Repartidor realiza la tarea solicitada y regresa a la lista de Repartidores disponibles.

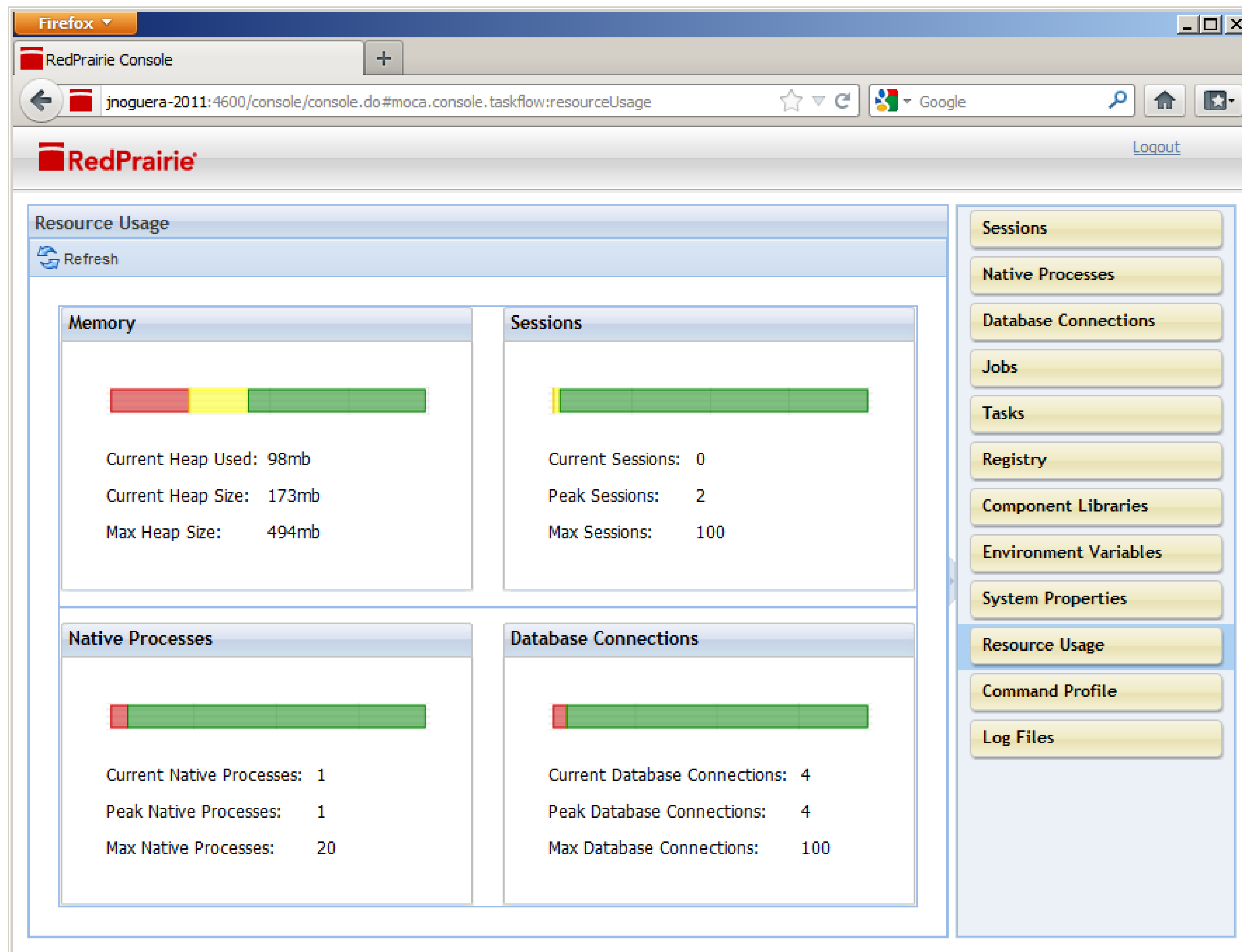
RedPrairie

Consola MOCA

- La consola MOCA es una utilería que ofrece información útil acerca del sistema.
- La información se divide en tres categorías:
  - > Información de MOCA server process.
  - > Información de la conexión del cliente.
  - > Tareas de fondo.

RedPrairie

MOCA Console



RedPrairie



MOCA: Elementos de un comando.

RedPrairie®

Elementos de un comando

- Nomenclatura
- Argumentos
- Datos Publicados
- Triggers

RedPrairie

Nomenclatura

- En términos de MOCA, todos los componentes de servicio son conocidos por un nombre de componente.
- El nombre del componente define una acción o instrucción bajo la cual el servidor actúa.

RedPrairie

Nomenclatura

- El nombre del componente es definido por una combinación de verbos/sustantivos.
  - El verbo define la acción; el sustantivo define el objetivo de la acción y puede incluir sintaxis gramática (an, a, for, the, etc...).
  - El verbo es una palabra sencilla, y el sustantivo puede ser múltiple.
- El verbo y los sustantivos están hechos para una clasificación de comandos más sencilla (p. e. "list" commands vs. "change" commands vs. "move" commands).

RedPrairie

Nomenclatura

- Todos los comandos y triggers personalizados deben ser nombrados como sigue para estandarizar la nomenclatura de comandos en el futuro.
  - <verbo> var <sustantivo(s)>, for VAR level changes
  - <verbo> usr <sustantivo(s)>, for USR level changes

- Algunos ejemplos son:
  - `move inventory`
  - `allocate var shipment location`
  - `rate usr shipment.`

RedPrairie

Argumentos

- Los comandos hechos por los clientes pasan argumentos y operadores.
  - > La transmisión de argumentos a los componentes es manejada por MOCA.
  - > No todos los argumentos necesitan ser pasados al componente.
  - > Los argumentos pueden ser requeridos u opcionales se pueden usar también valores constantes.

```
 move inventory
where lodnum = "LOD09"
  and dstloc = "X1034"
```


RedPrairie®

Argumentos - alias

- Algunos argumentos de comando tienen un nombre alternativo para hacer que la sintaxis del comando sea de uso más convencional.
  - > Un ejemplo más entendible del mismo comando sería:

```
 move inventory
where load = "LOD09"
  and destination = "X1034"
```

Published Data

- Los componentes pueden o no regresar datos a los clientes.
- El dato es publicado en formato de filas/columnas donde cada columna tiene un nombre. Al igual que los datos se representan en las hojas de cálculo.

RedPrairie

Triggers

- Además de los argumentos, los comandos también pueden tener triggers.
  - Los triggers pueden ser otros comandos o sentencias SQL.
- Los triggers siempre se ejecutan con un comando.
  - Si el trigger regresa un error, la transacción completa no se lleva a cabo, preservando la integridad de la transacción.
  - La configuración del trigger para el sistema se conserva en el mismo sistema y no es afectada por actualizaciones del producto.

RedPrairie



MOCA: Creación de comandos

Creación de comandos

- El añadir un nuevo comando al sistema DLx® consiste en un proceso de tres pasos:

1. Escribir la sintaxis apropiada y la documentación de soporte.
2. Añadir el comando al archivo de memoria de comandos disponibles.
3. Detener y reiniciar el sistema DLx® para que el nuevo comando sea accesible a todos los procesos de fondo.

RedPrairie

Escribir el Comando…

- Hay dos formas para crear comandos/triggers
  - > Administración de comandos de servidor
    - Aquí los comandos son creados con la ayuda de una aplicación en el cliente DLx®.
    - Los usuarios no necesitan saber la estructura del archivo solo los datos que contendrá cada comando.
  - > Creación manual de archivos (*.mcmd, *.mtrg)
    - Los comandos pueden ser creados usando cualquier editor de texto.
    - Aquí los usuarios deben saber la estructura del archivo del comando.



MOCA: Creación de comandos – Admón. de Comandos de Servidor

Admón de Comandos de Servidor

- La aplicación de Admón. de Comandos de Servidor es la ubicación central para ver, modificar, y crear comandos MOCA.
- La aplicación se divide en dos secciones principales, el árbol a la izquierda y la sección de detalles a la derecha.
- El árbol clasifica cada comando MOCA por secciones de aplicación.

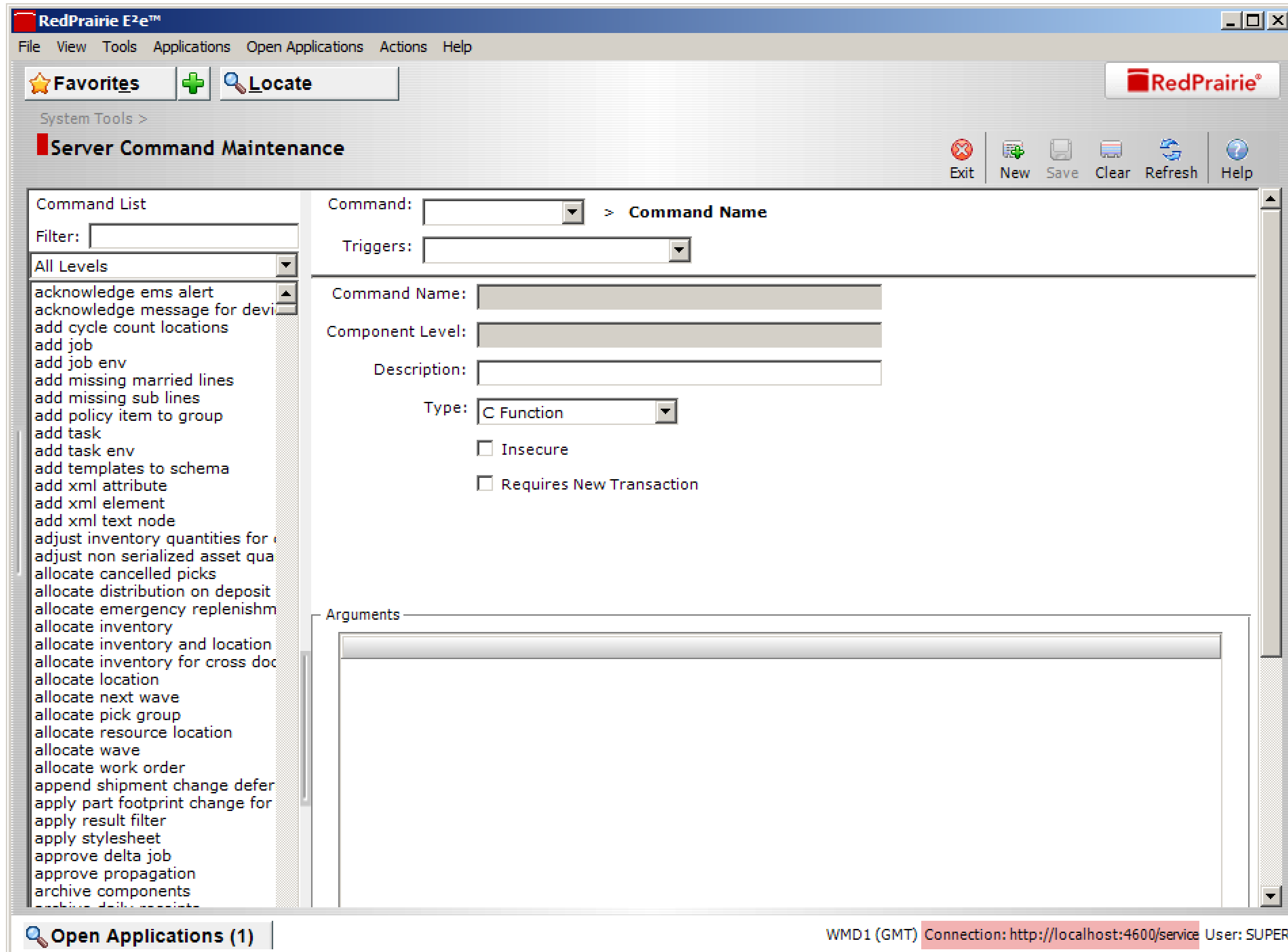
RedPrairie

Admón de Comandos de Servidor

- Para crear un nuevo comando, de clic en el botón nuevo en la barra de herramientas y la pantalla será habilitada para crear el nuevo comando.
  - > La pestaña de **Configuración** contiene el nombre, sintaxis, descripción del comando, etc.
  - > La pestaña de **Argumentos** lista todas las variables usadas en el nuevo comando MOCA.
  - > La pestaña **Triggers** liga el comando con el trigger correspondiente, si es que el comando ejecuta algún trigger.
  - > La pestaña **Documentación** muestra información del usuario acerca del comando y el contexto en el que se usa.

RedPrairie

Admón de Comandos de Servidor



RedPrairie



MOCA: Creación Manual del Archivo

Creando Comandos – Archivos *.MCMD

```
<command>
<name>list rfid gtin detail</name>
<description>List RFID GTIN detail.</description>
<type>Local Syntax</type>
<local-syntax>
<![CDATA[
[select *
   from rfid_tag_sernum
  where @+reference]
]]>
</local-syntax>
```

Información de funcionalidad del comando que incluye:

- Nombre
- Descripción
- Tipo de Sintaxis

```
<documentation>
<remarks>
<![CDATA[
 This command lists existing RFID serial numbers.
]]>
</remarks>
<exception value="eOK">
 The command completed successfully.
</exception>
</documentation>
</command>
```

Información de documentación del comando que incluye:

- Comentarios
- Descripción
- Ejemplos
- Datos devueltos

RedPrairie

Creando Comandos – Archivos *.MCMD

```
<command>
<name>list rfid gtin detail</name>
<description>List RFID GTIN detail.</description>
<type>Local Syntax</type>
<local-syntax>
<![CDATA[
[select *
   from rfid_tag_sernum
  where @+reference]
]]>
</local-syntax>


<documentation>
<remarks>
<![CDATA[
 This command lists existing RFID serial numbers.
]]>
</remarks>
<exception value="eOK">
 The command completed successfully.
</exception>
</documentation>
</command>
```

- Nombre del comando
  - El nombre del comando es lo que es referenciado o llamado por el sistema DLx® para ejecutar el código que contiene
  - El nombre es escrito como una descripción en ingles.
  - Los nombres son escritos en un formato de <verbo><nombre>.

RedPrairie

Creando Comandos – Archivos *.MCMD

```
<command>
<name>list rfid gtin detail</name>
<description>List RFID GTIN detail.</description>
<type>Local Syntax</type>
<local-syntax>
<![CDATA[
[select *
   from rfid_tag_sernum
  where @+reference]
]]>
</local-syntax>


<documentation>
<remarks>
<![CDATA[
 This command lists existing RFID serial numbers.
]]>
</remarks>
<exception value="eOK">
 The command completed successfully.
</exception>
</documentation>
</command>
```

- Descripción del comando
  - La descripción solo es usada para informar a los usuarios el propósito del comando.
  - No es usada para la ejecución del comando.
  - La descripción del comando debería contener más información que sólo el nombre del comando.

Creando Comandos – Archivos *.MCMD

```
<command>
<name>list rfid gtin detail</name>
<description>List RFID GTIN detail.</description>
<type>Local Syntax</type>
<local-syntax>
<![CDATA[
[select *
   from rfid_tag_sernum
  where @+reference]
]]>
</local-syntax>


<documentation>
<remarks>
<![CDATA[
 This command lists existing RFID serial numbers.
]]>
</remarks>
<exception value="eOK">
 The command completed successfully.
</exception>
</documentation>
</command>
```

- Tipo de comando
  - > Determina que tipo de sintaxis de comando será ejecutada.
  - > Los tipos incluyen:
    - Funciones C
    - Funciones C simples
    - Métodos COM
    - Local Syntax

Creando Comandos – Archivos *.MCMD

```
<command>
<name>list rfid gtin detail</name>
<description>List RFID GTIN detail.</description>
<type>Local Syntax</type>
<local-syntax>
<![CDATA[
[select *
   from rfid_tag_sernum
  where @+reference]
]]>
</local-syntax>


<documentation>
<remarks>
<![CDATA[
 This command lists existing RFID serial numbers.
]]>
</remarks>
<exception value="eOK">
 The command completed successfully.
</exception>
</documentation>
</command>
```

- Sintaxis del Comando
  - > Local syntax puede ejecutar:
    - Sentencias SQL
    - Sintaxis MOCA
    - Otros comandos MOCA

RedPrairie

Recargar Memoria – Windows

1. Abra una ventana CMD y vaya a:

{Drive}\RedPrairie\<instance name>\MOCA\bin

```
Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\mstraka>cd \redprairie\dcs_dev\moca\bin

C:\RedPrairie\dcs_dev\MOCA\bin>_
```

RedPrairie

Recargar Memoria – Windows

2. Ahora necesitamos crear nuestro archivo de ambiente ejecutando:

```
servicemgr /dump /env=<instance_name>
```

Command Prompt

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\mstraka>cd \redprairie\dcs_dev\moca\bin

C:\RedPrairie\dcs_dev\MOCA\bin>servicemgr /dump /env=dcs_dev

C:\RedPrairie\dcs_dev\MOCA\bin>
```

RedPrairie

Recargar Memoria – Windows

3. Ahora cargaremos el ambiente ejecutando:

   `env`

```
Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\mstraka>cd \redprairie\dcs_dev\moca\bin

C:\RedPrairie\dcs_dev\MOCA\bin>servicemgr /dump /env=dcs_dev

C:\RedPrairie\dcs_dev\MOCA\bin>env

C:\RedPrairie\dcs_dev\MOCA\bin>set MOCA_ENVNAME=dcs_dev

C:\RedPrairie\dcs_dev\MOCA\bin>set CLASSPATH=C:\RedPrairie\dcs_dev\MOCA\javalib/
hibernate3.jar;C:\RedPrairie\dcs_dev\MOCA\javalib/jtds.jar;C:\RedPrairie\dcs_dev
\MOCA\javalib/junit.jar;C:\RedPrairie\dcs_dev\MOCA\javalib/ojdbc14.jar;C:\RedPra
irie\dcs_dev\MOCA\javalib/sftp.jar;C:\RedPrairie\dcs_dev\MOCA\lib/moca.jar;C:\Re
dPrairie\dcs_dev\MCS\lib/mcs.jar;C:\RedPrairie\dcs_dev\SAL\lib/sal.jar

C:\RedPrairie\dcs_dev\MOCA\bin>set DCSDIR=C:\RedPrairie\dcs_dev\DCS

C:\RedPrairie\dcs_dev\MOCA\bin>set LESDIR=C:\RedPrairie\dcs_dev\LES

C:\RedPrairie\dcs_dev\MOCA\bin>set LOCALE_ID=US_ENGLISH

C:\RedPrairie\dcs_dev\MOCA\bin>set MCSDIR=C:\RedPrairie\dcs_dev\MCS

C:\RedPrairie\dcs_dev\MOCA\bin>set MOCADIR=C:\RedPrairie\dcs_dev\MOCA

C:\RedPrairie\dcs_dev\MOCA\bin>set MOCARPTDIR=C:\RedPrairie\dcs_dev\MOCARPT

C:\RedPrairie\dcs_dev\MOCA\bin>set PATH=C:\RedPrairie\dcs_dev\LES\scripts;C:\Red
Prairie\dcs_dev\LES\bin;C:\RedPrairie\dcs_dev\MOCA\bin;C:\RedPrairie\dcs_dev\MOC
A\scripts;C:\RedPrairie\dcs_dev\SEAMLES\bin;C:\RedPrairie\dcs_dev\SEAMLES\script
s;C:\RedPrairie\dcs_dev\MCS\bin;C:\RedPrairie\dcs_dev\MCS\scripts;C:\RedPrairie\
dcs_dev\SAL\bin;C:\RedPrairie\dcs_dev\SAL\scripts;C:\RedPrairie\dcs_dev\MOCARPT\
bin;C:\RedPrairie\dcs_dev\MOCARPT\scripts;C:\RedPrairie\dcs_dev\DCS\bin;C:\RedPr
airie\dcs_dev\DCS\scripts;C:\Perl\bin\;C:\PROGRAM FILES\THINKPAD\UTILITIES;C:\WI
NDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Program Files\ATI Technolo
gies\ATI Control Panel;C:\Program Files\PC-Doctor for Windows\services;C:\PROGRA
~1\ABSOLU~1;C:\Program Files\Symantec\pcAnywhere\;C:\Program Files\Microsoft SQL
 Server\80\Tools\BINN;C:\Program Files\QuickTime\QTSystem\;C:\Program Files\Java
\jre1.5.0_06\bin;C:\Program Files\Java\jre1.5.0_06\bin\client

C:\RedPrairie\dcs_dev\MOCA\bin>set REPORTSDIR=C:\RedPrairie\dcs_dev\LES\files\re
ports

C:\RedPrairie\dcs_dev\MOCA\bin>set SALDIR=C:\RedPrairie\dcs_dev\SAL

C:\RedPrairie\dcs_dev\MOCA\bin>set SL_INHIBIT_ORA_OBJ=true

C:\RedPrairie\dcs_dev\MOCA\bin>set SLDIR=C:\RedPrairie\dcs_dev\SEAMLES

C:\RedPrairie\dcs_dev\MOCA\bin>_
```

RedPrairie®

Recargar Memoria – Windows

4. Finalmente ejecute el comando:

```
mbuild
```

```
C:\RedPrairie\dcs_dev\MOCA\bin>set SL_INHIBIT_ORA_OBJ=true

C:\RedPrairie\dcs_dev\MOCA\bin>set SLDIR=C:\RedPrairie\dcs_dev\SEAMLES

C:\RedPrairie\dcs_dev\MOCA\bin>mbuild

MBUILD 2005.2.5 - Production on Fri Jun 09 13:48:23 2006

Copyright (c) 2002-2005 RedPrairie Corporation.  All rights reserved.


Reading command repository...
Writing commands memory file...


Built Component Levels:    62
              Commands:  4634
              Triggers:   462
             Arguments: 13949
          Local Syntax:  3077

    Total Elapsed Time: 17034 ms

C:\RedPrairie\dcs_dev\MOCA\bin>_
```

RedPrairie

Recargar Memoria – Windows

- Después de que el archivo de memoria ha sido reconstruido los servicios deben reiniciarse para que puedan acceder a los nuevos comandos.

```
Command Prompt

Reading command repository...
Writing commands memory file...

Built Component Levels:    62
              Commands:  4634
              Triggers:   462
             Arguments: 13949
          Local Syntax:  3077

    Total Elapsed Time: 17034 ms

C:\RedPrairie\dcs_dev\MOCA\bin>net stop moca.dcs_dev
The RedPrairie (dcs_dev) service is stopping.
The RedPrairie (dcs_dev) service was stopped successfully.

C:\RedPrairie\dcs_dev\MOCA\bin>net start moca.dcs_dev
The RedPrairie (dcs_dev) service is starting.
The RedPrairie (dcs_dev) service was started successfully.

C:\RedPrairie\dcs_dev\MOCA\bin>
```

RedPrairie



MOCA: Reemplazo de Variables

Remplazo de variables – @variable

@variable

- Remplazado por el valor de variable
- El remplazo ocurre solo sí el operador usado para ingresar el valor de variable en la cláusula where es ‘=’

RedPrairie

@variable

```
@variable

[SELECT *
   FROM ord
  WHERE ordnum = @ordnum
    AND wh_id  = @wh_id]
```

```
SELECT *
  FROM ord
 WHERE ordnum = 'TESTORD01'
   AND wh_id  = 'WMD1'
```

@variable

```
@variable

[SELECT *
   FROM ord
  WHERE ordnum = @ordnum
    AND wh_id  = @wh_id]
```

```
SELECT *
  FROM ord
 WHERE ordnum = 'TESTORD01'
   AND wh_id  = NULL
```

RedPrairie

Reemplazo de variables – @+*variable*

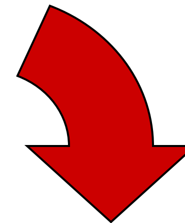
```
@+variable
```

- **Reemplazado por variable=value**
- **Sí la variable no es encontrada, es reemplazada por 1=1**
- **Sí la variable fue puesta en la cláusula where con otro operador diferente de "=", se usa ese operador.**

RedPrairie

@+variable

```
[SELECT *
   FROM ord
  WHERE @+ordnum
    AND @+wh_id]
```

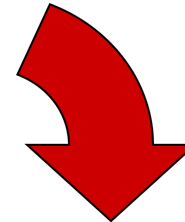


```
SELECT *
  FROM ord
 WHERE ordnum = 'TESTORD01'
   AND wh_id  = 'WMD1'
```

@+variable

@+variable

[SELECT *
FROM ord
WHERE @+ordnum
AND @+wh_id]



```
SELECT *
  FROM ord
 WHERE ordnum = 'TESTORD01'
   AND 1 = 1
```

Reemplazo de variables – *@%variable*

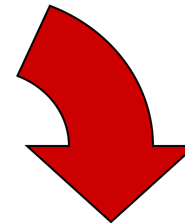
```
@%variable
```

- **Si el valor de la variable contiene los caracteres '_' o '%', es reemplazado por variable like value, de otra manera remplazado usando el mismo formato que el usado para @+variable.**

@%variable

@%variable

[SELECT *
FROM ord
WHERE @%ordnum
AND @%wh_id]

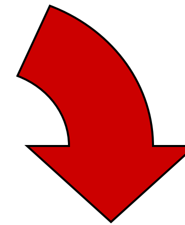


```
SELECT *
  FROM ord
 WHERE ordnum = 'TESTORD01'
   AND wh_id  = 'WMD1'
```

@%variable

@%variable

[SELECT *
FROM ord
WHERE @%ordnum
AND @%wh_id]



```
SELECT *
  FROM ord
 WHERE ordnum = 'TESTORD01'
   AND 1 = 1
```

RedPrairie

@%variable

@%variable

```
[SELECT *
   FROM ord
  WHERE @%ordnum
    AND @%wh_id]
```

```
SELECT *
  FROM ord
 WHERE ordnum = 'TESTORD01'
   AND wh_id LIKE 'WM%'
```

RedPrairie

Reemplazo de variables – @*

```
@*
```

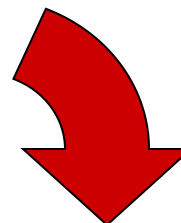
- **Reemplazado por todas las variables en la cláusula where de la invocación actual del comando, en orden, en el mismo formato que el usado para @+variable.**

RedPrairie

@*

@*

[SELECT *
  FROM ord
 WHERE @%ordnum
   AND @*]



```
SELECT *
  FROM ord
 WHERE ordnum = 'TESTORD01'
   AND wh_id  = 'WMD1'
```

RedPrairie

Reemplazo de variables – *@@variable*

`@@Variable`

- **Reemplazado por el valor de la variable de ambiente variable.**

@@variable

@@variable

[SELECT *
FROM users_view
WHERE usr_id = @@usr_id]

```
SELECT *
   FROM users_view
  WHERE usr_id = 'SUPER'
```

RedPrairie

Reemplazo de variables – @?

```
@?
```

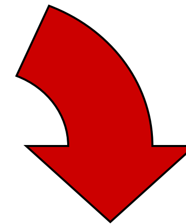
- **Reemplazada por el estatus de terminación del último comando ejecutado.**

RedPrairie

@?

@?

```
[SELECT *
   FROM ord_dtl] catch(@?)
  |
  publish data
    where err_code = @?
```



```
Publish data
Where err_code = '0'
```

Reemplazo de variables – @*!*

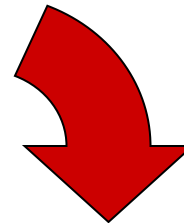
```
@!
```

- **Reemplazado por el mensaje de error del último comando ejecutado**

@!

@!

```
[SELECT *
   FROM ord_dtl] catch(@?)
    |
    publish data
      where err_sts = @!
```



```
Publish data
Where err_sts = 'Success'
```



MOCA: Sintáxis Local

Sentencias SQL

- Los comandos escritos entre corchetes son considerados SQL.
- MOCA pasa sentencias SQL directamente al motor de base de datos después de efectuar el reemplazo de la variable.

[sql statement]

RedPrairie

Sentencias SQL

```
[SELECT ord_num,
        ord_typ_txt,
        line_num,
        ord_qty
   FROM ord
  WHERE ord_num     = 'TESTORD01'
    AND ord_typ_txt = 'CUST']
```

RedPrairie

Separadores de Comandos

- El punto y coma permite ejecutar una secuencia de comandos y realizar un commit si todos los comandos son completados exitosamente o un rollback si cualquiera de los comandos falla.
  - Cada uno de los comandos separados por punto y coma se ejecuta independientemente de cualquier otro, es decir, no tienen datos devueltos por el comando anterior.

```
command; command
```

RedPrairie

Comando Pipe

- El pipe permite ejecutar un flujo de comandos.
- Cada uno de los comandos separados por pipes tienen acceso a publicar datos y argumentos de comandos anteriores.

```
command | command
```

Comando Pipe

Publish data

where ord_num = ‘TESTORD01’

and   ord_typ_txt = ‘CUST’

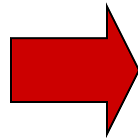
|

[SELECT *

FROM ord_dtl

WHERE @+ord_num

AND @+ord_typ_txt]



```
SELECT *
  FROM ord_dtl
 WHERE ord_num = 'TESTORD01'
   AND ord_typ_txt = 'CUST'
```

RedPrairie

Concatenación de comandos

- El ampersand concatena el resultado generado por un grupo de comandos dentro de un flujo.

```
command & command
```

RedPrairie®

Agrupación de comandos

- Pueden ser usados para agrupar comandos, permitiendo ejecutar comandos independientemente de algún otro que participe en el flujo por medio de un pipe.

```
command | { command; command }
```

RedPrairie

Redireccionamiento

- El conjunto de resultados puede ser puesto en una variable usando redireccionamiento.

```
{command & command} >> varname
|
if (rowcount (@varname) != 0)
   command
```


RedPrairie

Redireccionamiento

```
[select *
   from table] >> record_set
|
if (rowcount (@record_set) >= 250)
{
     [select *
        from table
       where rownum <= 250]
}
else
{
     [select *
        from table]
}
```

RedPrairie

Condicionales

- Los comandos pueden ser condicionalmente ejecutados usando una tendencia if-else.

```
If (expr)
{
  command1
}
else
{
  command2
}
```

Condicionales

```
if (@full_ship_flg = 1)
{
  [SELECT *
     FROM ord_dtl
    WHERE ord_num = @+ord_typ_txt
      AND ord_qty + adj_qty = shp_qty]
}
else
{
  [SELECT *
     FROM ord_dtl
    WHERE ord_num = @+ord_typ_txt]
}
```

RedPrairie

Excepciones

- Todos los comandos que no terminen exitosamente lanzan excepciones.
  - > Las excepciones pueden ser atrapadas usando la sentencia catch.
  - > Las excepciones que son atrapadas permiten continuar la ejecución y causan que el código de error del comando sea cambiado a 0 para que la transacción de la base de datos sea efectuada.
  - > Las excepciones que no sean atrapadas causan que la ejecución del comando actual sea detenida inmediatamente.

```
command catch(errorcode1, [errorcode2, […]])
```

RedPrairie®

Ejecución remota de comandos

- Un comando puede ser executado en un servidor remoto MOCA usando la palabra remote.
  - El formato esperado es el siguiente:

`hostname:port`.

```
remote(system) command
```

```
remote ('wia4at01:4700#50') list users
```


RedPrairie

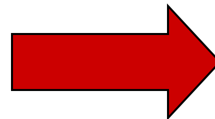


MOCA: Reemplazo de Variables – Tipos de Cast

Reemplazo de variables Tipos de Cast – :raw

- Usando una variable de la forma `@variable:raw` indicara a MOCA que escriba el contenido literal de esa variable.
- Esto es útil cuando el valor de `variable` es una lista de cadenas de caracteres.

```
Publish data
where @order_by = 'ord_num, ord_typ_txt'
|
 [SELECT *
    FROM ord
ORDER BY @order_by:raw]
```



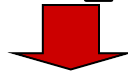
```
 [SELECT *
    FROM ord
ORDER BY ord_num,
         ord_typ_txt]
```

RedPrairie

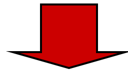
Reemplazo de variables Tipos de Cast – :date

- Usando una variable de la forma `@variable:date` indicara a MOCA que escriba el contenido de la variable con formato de fecha.
- Esto es útil cuando el valor de `variable` es una fecha

```
publish data where @rpt_dat = '20050101000000'
|[SELECT * FROM ord WHERE rpt_dat = @rpt_dat:date]
```



```
select * from ord where adddte = TO_DATE('20050101010203')
```



```
select * from ord
 where adddte = CONVERT(DATETIME, '2005-01-01 01:02:03')
```

RedPrairie



MOCA: Operadores

Operadores

- Operadores

+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo

- Operadores de cadena de caracteres

||	Concatena dos cadenas

Operadores

- Operadores lógicos

`!`	Revierte las expresiones lógicas.
`AND`	Combina 2 o más expresiones lógicas: TRUE si todo es verdadero; sino FALSE.
`OR`	Combina 2 o más expresiones lógicas: TRUE si cualquiera es verdadero; sino FALSE.

RedPrairie®

Operadores

- Operadores de comparación

`=`	Igualdad
`!= <>`	diferencia
`> >=`	Mayor que, Mayor o igual a.
`< <=`	Menor que, Menor o igual a
`[NOT] LIKE`	Coincide [No coincide] con un patrón específico.
`IS[NOT] NULL`	[no] nulo.



MOCA: Funciones

Funciones – decode()

`decode(expr, search, result[, search, result, default])`

- Para evaluar esta expresión MOCA compara `expr` con cada valor de búsqueda uno por uno.
- Si `expr` es igual a una búsqueda, MOCA regresa el resultado correspondiente.
- Si ninguna coincidencia es encontrada, MOCA regresa `default,` o, si `default` es ignorado, regresa un valor nulo.

RedPrairie

Funciones – nvl()

```
nvl(expr, if-null-expr)
```

- Si expr esta determinada a ser NULL. La expresión representada por if-null-expr es devuelta, sino expr es devuelto.

```
 list orders
where act_dt >= nvl(@beg_act_dt,act_dt)
  and act_dt <= nvl(@end_act_dt,act_dt)
```

RedPrairie



MOCA: Wrappers

Wrappers

- Wrappers ofrecen otra manera de extender la funcionalidad de un componente particular.
- Un wrapper es un componente que usa MOCA para manejar múltiples niveles de componentes del mismo nombre.
- Puede realizar alguna función y luego invocar a la siguiente versión de nivel de componente, si tiene el mismo nombre.

RedPrairie

Wrappers

```
if (@special_pack_list = 'Y')
   produce special packing list
else
   ^produce packing list
```



MOCA: Comando Útiles

Comandos Útiles

```
list user tables
```

> Lista todas las tablas

```
list table columns
where table = 'table_name'
```

> Lista todas las columnas en una tabla

```
list tables with column
where column = 'column_name'
```

> Lista todas las tablas que contengan la columna `column_name`.

RedPrairie®

Comandos Útiles

```
publish data where a=1 and b=2
```

> Este comando publica todos los argumentos en su cláusula where como un conjunto de resultados

```
list library versions
```

- Este comando lista todas las versiones de las librerías.

RedPrairie

Comandos Útiles

```
 list column comment
where table = ‘table name'
  and column = ‘column name'
```

> Este comando lista todos los comentarios asociados a una columna en la tabla

```
 list common columns on tables
where table1 = ‘first table name'
  and table2 = ‘second table name'
```

> Este comando lista las columnas compartidas entre 2 tablas

```
 list table comment
where table = ‘table name‘
```

> Esto listará los comentarios para la tabla seleccionada

RedPrairie®

Comandos Útiles

```
list database locks
```

- Este comando lista cualquier bloqueo que pueda existir en la base de datos

```
list primary key for table
where = ‘table name'
```

- Este comando lista todas las llaves primarias de una tabla|

```
list user tables |
list table comment
 where table = @table_name catch(@?)
```

Lista todos los comentarios de todas las tablas del sistema.

RedPrairie

Comandos Útiles

`to_char`

- > Esta función MOCA es usada para convertir fechas a caracteres.

`to_date`

- > Esta función MOCA es usada para convertir cadenas a fechas.

`to_number`

- > Esta función MOCA es usada para convertir cadenas de caracteres a números.

RedPrairie

Comandos Útiles

```
 [select sysdate dt
    from dual] |
publish data where x = to_char(@dt,
   'YYYYMMDD')

 [select to_char(sysdate,'YYYYMMDD') dt_str
    from dual] |
publish data where x = to_date(@dt_str,
   'YYYYMMDD')

 [select '1123' num
    from dual] |
publish data where x = to_number(@num,
   '99V99')
```

RedPrairie

Comandos Útiles

```
send email
```

- Este comando envía un e-mail al destinatario dado.
- Si un documento es dado, su contenido es enviado en el e-mail.
- Si uno o más archivos adjuntos son dados, son enviados en el e-mail.

```
put ftp
```

- Este comando envía un archivo a otro host vía ftp.

RedPrairie®

Comandos Útiles

```
convert list
```

> El comando toma una cadena, un separador, y un tipo de conversión y separa cada palabra de la cadena dada usando el separador dado, construyendo una nueva lista del tipo de conversión dado.

- Si el tipo de conversión es una 'L' (list), la cadena dada es dividida, con cada fila en el conjunto de resultados siendo una palabra de la cadena.
- Si el tipo de conversión es una 'S', cada palabra en la cadena dada se encierra entre comillas sencillas.
- Si el tipo de conversión es una 'D', cada palabra en la cadena dada es encerrada entre comillas.
- Si no se pasa un separador, se usa la coma.

RedPrairie

Comandos Útiles

```
convert list
 where string = 'a,b,c,d'
   and type = 'L'

   retstr  count
   ------  -----
   a       1
   b       2
   c       3
   d       4
```